10-1-2015

An Incremental Phylogenetic Tree Algorithm Based on Repeated Insertions of Species

Peter Revesz
University of Nebraska-Lincoln, prevesz1@unl.edu

Zhiqiang Li
University of Nebraska-Lincoln, zli13@unl.edu

An Incremental Phylogenetic Tree Algorithm Based on Repeated Insertions of Species

Peter Z. Revesz, Zhiqiang Li

***Abstract*—**In this paper, we introduce a new phylogenetic tree algorithm that generates phylogenetic trees by repeatedly inserting species one-by-one. The incremental phylogenetic tree algorithm can work on proteins or DNA sequences. Computer experiments show that the new algorithm is better than the commonly used UPGMA and Neighbor Joining algorithms.

***Keywords*—**Data structure, Distance matrix, Phylogenetic tree, Protein.

I. Introduction

CURRENT phylogenetic tree construction algorithms[1]-[3], [5], [9], [11] are not incremental and have to be rerun from the beginning whenever a new species is added to the database. Moreover, a rerun from the beginning is necessary even if the new species is aligned with the already used species. In this paper, we develop an incremental algorithm that inserts new species one-by-one into a growing phylogenetic tree.

Our inspiration for such an incremental phylogenetic algorithm is the way biologists usually classify any newly discovered species. Starting from the root node of the existing classification tree, the newly discovered species is compared with existing species and always an appropriate branch is chosen to go one level down in the classification hierarchy. Eventually we reach one of the existing species, which is the closest relative. It is next to that nearest relative where the new species is normally inserted.

Our aim is to develop a computer algorithm that uses the above paradigm but works with both DNA sequences and proteins. As the genomes of a growing number of species are sequenced and become part of DNA and protein databases [4], [12], molecular biology increasingly augments, although not completely replaces, morphological considerations.

Reliable phylogenetic tree constructions are needed for a diverse set of studies, including theoretical studies on the rate of evolution in various phyla [10] and applied studies aimed at developing medical diagnosis methods [6] and pharmaceutical development.

Our algorithm has two main benefits compared to previous algorithms:

1) *Faster* because it can be used incrementally if the new sequence is aligned with the other sequences.
2) Generates *more accurate* phylogenetic trees as indicated by the computer experiments presented in Section 4.

This paper is organized as follows. Section II presents some related work. Section III describes the incremental phylogenetic tree algorithm. Section IV presents some experimental results. Finally Section V gives some conclusions and directions for future work.

II. Related work

The *UPGMA* [11] and the *Neighbor Joining* [9] algorithms are commonly used and familiar to most users. The maximum likelihood method is also well known, although it seems less frequently used that UPGMA and Neighbor Joining in practice because it requires more computational time. All of these algorithms are reviewed in textbooks, such as [1]-[3].

Revesz [5] introduced the *Common Mutations Similarity Matrix* algorithm, which has $O(n^3)$ time complexity, where n is the number of sequences. We briefly review this algorithm as a related work, which will also be used in the experimental results section of this paper.

Table 1 below shows seven DNA sequences, $S_1 \ldots S_7$, each with a length fifteen nucleotides displayed by groups of five nucleotides per column.

Table 1 Seven input DNA sequences and a common ancestor μ

S1	AGCTA	CTAGT	AATCA
S2	AGCTA	CGAGT	AATCA
S3	ATCCA	CTAGT	ACACT
S4	ATCCA	CTAGT	ATACT
S5	CGGTA	TTTGT	AAGCT
S6	CGGTT	CATCA	AATGC
S7	AGGTA	CTTGA	AATCC
μ	AGCTA	CTAGT	AATCT

Let Si[k] denote the kth nucleotide of Si. The *Hamming distance* between two DNA sequences Si and Sj each with length n, denoted δ (Si, Sj), is defined as the number of corresponding nucleotide pairs that are different, that is, $\Sigma 1 \leq k \leq n$ Si [k] ≠ Sj[k]. μ is the common ancestor of seven sequences.

Evolutionary tree construction algorithms generally start

Peter Z. Revesz is with the Department of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, NE 68588, USA(revesz@cse.unl.edu).

Zhiqiang Li is with the Department of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, NE 68588, USA (zli@cse.unl.edu).

from a *Hamming distance matrix* to recursively combine pairs of sequences (rows and columns) until only a single combined sequence remains. For example, the UPGMA (unweighted pair group method with arithmetic mean) [11] method would always search for the closest pairs to combine. When several pairs are equally distant, then an arbitrary choice is made. In this case, the closest pairs are S1 and S2 and S3 and S4 because δ (S1, S2) = 1 and δ (S3, S4) = 1. The Neighbor Joining [9] method is a more sophisticated and commonly used method that is also based on distance matrices.

Instead of distance matrices, Revesz [5] introduced a *common mutations similarity matrix (CMSM)*. The motivation behind looking for common mutations is that in practice rare but shared features, such as rare mutations, often provide useful markers of similarity among a set of closely related items. Moreover, if mutations are rare, then it may be more efficient to count their occurrences than finding the Hamming distances for long sequences. Assuming that the seven DNA sequences in Table 1 are related, we can find the most likely common ancestor sequence, denoted μ, as the mode of each column. If there is no most frequent nucleotide in a column, then we arbitrarily chose one of the most frequent nucleotides in it.

The Common Mutations Similarity Matrix (CMSM) algorithm records for each pair of sequences the mutations that they share in common with respect to a global average μ, which is taken as the most likely common ancestor sequence.

Example 1. Given seven nucleotide sequences in Table 1 below (rows S1 to S7 where the sequences are displayed in groups of five), the common ancestor sequence μ is calculated in [5] as the most frequent in each column.

Alternatively, if S1 to S7 are considered amino acid sequences where A, C, G and T now stand for the amino acids Alanine, Cysteine, Glycine and Threonine, respectively, then the common ancestor sequence μ can be defined as in each column as the amino acid x out of the set S of twenty amino acids used in most proteins such that x is overall closest to the set of amino acids in that column. We make this statement more precise below using as an example the PAM250 amino acid similarity matrix. Let

$$\text{PAM250}[\text{AminoAcid1}, \text{AminoAcid2}] = \text{a} \qquad (1)$$

denotes that AminoAcid1 and AminoAcid2 have a similarity score of a. For example, PAM250 [A, G] = 1 means that Alanine and Glycine are slightly similar to each other. Then for the ith column,

$$\mu[i] = x \in S \qquad (2)$$

such that

$$\sum_{j=1}^{7} PAM250[Si[j], x] \qquad (3)$$

is maximum.

For example, we can see that the value of μ[1] changed from A to C because C is the amino acid that is overall closest to the each of the amino acids in the first column.

Table 2 Common ancestor μ from the new algorithm

S1	AGCTA	CTAGT	AATCA
S2	AGCTA	CGAGT	AATCA
S3	ATCCA	CTAGT	ACACT
S4	ATCCA	CTAGT	ATACT
S5	CGGTA	TTTGT	AAGCT
S6	CGGTT	CATCA	AATGC
S7	AGGTA	CTTGA	AATCC
μ	CGCCA	CTTGT	AATCC

It can be assumed that in each sequence Si those amino acids (or nucleotides) that do not match the corresponding amino acid (or nucleotide) in μ were mutated at some point during evolution. Intuitively, the more common mutations two sequences Si and Sj share, the closer they are likely to be in an evolutionary tree. For the above set of sequences, the common mutations similarity matrix is shown in Table 3:

Table 3 Initial CMSM matrix

	S1	S2	S3	S4	S5	S6	S7
S1	0	4	2	2	1	1	2
S2	4	0	2	2	1	1	2
S3	2	2	0	5	1	0	1
S4	2	2	5	0	1	0	1
S5	1	1	1	1	0	2	2
S6	1	1	0	0	2	0	3
S7	2	2	1	1	2	3	0

According to the common mutations similarity matrix, the closest pair of sequences is S3 and S4. Hence these will be merged. When we merge two sequences Si and Sj, in the merged sequence the kth element will be equal to the amino acid (or nucleotide) in the two sequences if Si[k] = Sj[k] and will be equal to μ[k] otherwise. Hence the matrix of sequences will be updated as Table 4:

Table 4 The updated sequences

S1	AGCTA	CTAGT	AATCA
S2	AGCTA	CGAGT	AATCA
S34	ATCCA	CTAGT	AAACT
S5	CGGTA	TTTGT	AAGCT
S6	CGGTT	CATCA	AATGC
S7	AGGTA	CTTGA	AATCC
μ	CGCCA	CTTGT	AATCC

For example, since S3[12] = C ≠ T = S4[12], by the above merging rule S34[12] = μ[12] = A.

After the merge, the common mutations matrix needs to be recalculated. The merge does not change μ, but the entries in the common mutations similarity matrix that are related to the newly merged sequence S34 need to be calculated. The values for S3 and S4 should be deleted. In this case, Table 5 shows the updated common mutation matrix.

ISBN: 978-1-61804-350-4

Table 5 The updated CMSM matrix

	S1	S2	S34	S5	S6	S7
S1	0	4	2	1	1	2
S2	4	0	2	1	1	2
S34	2	2	0	1	0	1
S5	1	1	1	0	2	2
S6	1	1	0	2	0	3
S7	2	2	1	2	3	0

Now the closest pair is S1 and S2 with a value of 4 common mutations. Hence those two will be merged next. The merging will continue until there is only one sequence left. The CMSM evolutionary tree algorithm can be summarized as follows.

Table 6 The CMSM algorithm

```
ALGORITHM CMSM (S1…Sn, n)
1 Form n clusters of sequences, each with a single sequence.
2 Find the putative common ancestor μ of the sequences.
3 Construct a graph T with a node for each n cluster and for μ.
4 While (there is more than one cluster)
5       Find the common mutations similarity matrix.
6       If (exist distinct Si and Sj with some common mutations)
7          Merge a closest distinct Si and Sj pair into a new
8          cluster Sij and create a node for Sij.
9          Connect the nodes for Si and Sj with parent node Sij.
10      Else
11            Connect the remaining clusters' nodes to parent μ.
11            Return T.
12 Return T.
```

Note: Alternatively, instead of only recording the values, the actual set of common mutations can be put into each entry of the common mutations similarity matrix. Clearly, the cardinality of the sets in the second representation determines the numerical values in the first representation.

III. INCREMENTAL PHYLOGENETICS BY REPEATED INSERTIONS

A. A New Phylogenetic Tree Algorithm

Suppose that we have n number of amino acid sequences S1, . . ., Sn. The sequences and the number n are the inputs to the following algorithm that constructs an evolutionary tree by repeated addition of new species that are represented by the amino acid sequences. We call the new algorithm IPRI (incremental phylogenetic by repeated insertions).

In the algorithm, the closest pair can be found by minimum Hamming distance if the sequences are DNA or RNA strings. If the sequences are proteins, then the closest pair can be found by using a PAM or a BLOSUM substitution matrix. The running time is $O(n^2m)$ where m is the length of the sequences because there are n insertions, and each insertion requires n comparisons between two strings of length m.

Table 7 The IPRI algorithm

```
ALGORITHM IPRI(S1…Sn, n)
1 Create an independent node Nk for each sequence Sk.
2 Let N = { Nk : 1 ≤ k ≤ n }
3 Find the closest pair of nodes Nj and Nj.
4 Create a tree T with root R, left child Ni and right child Nj.
5 N = N \ {Ni, Nj}
6 While (N is not empty)
7       Find the closest pair of nodes Ni ∈ N and Mj ∈ T.
8       If (Mj is not the root of T)
9        P = parent of Mj.
10            Delete P as a parent of Mj.
11            Create a node R.
12            Make P the parent of R.
13            Make R the parent of Ni and Mj.
14       Else
15            Create a node R.
16            Make R the parent of Ni and Mj.
17      N = N \ {Ni}.
18 Return T.
```

IV. EXPERIMENTAL RESULTS

We compared the algorithms on simulated evolutionary data as follows. We assumed that the original protein consists of a chain of one thousand Alanine amino acids. We mutated this original string two ways to generate to children. Both children were generated by first randomly selecting one percent of the amino acids. Then we changed the selected amino acids to one of the twenty amino acids. That is, each of the selected amino acids had a five percent chance of remaining A and ninety five percent chance of changing into another amino acid, with five percent chance of changing into C, five percent chance of changing into D and so on.

Next both of the children were further mutated to generate four grandchildren of the original protein. Then we general additional levels of the tree so that after N levels we had 2^N leaves.

With the above process of evolutionary tree generation, two siblings can be expected to differ from each other on twenty amino acids. Two first cousins can be expected to differ from each other on forty amino acids. Two seconds can be expected to differ from each other on sixty amino acids, and so on.

We ran ten tests on evolutionary trees with height four (and sixteen leaves). We implemented the CMSM and the IPRA algorithms in MATLAB. We used ClustalW2's implementation of the UPGMA and NJ algorithms. We chose on the ClustalW2 website the default parameters, that is, a gap open penalty of 10, a gap extension penalty of 0.2, and a maximum gap distance of 5. The results can be summarized in the Table 8, where ``Perfect'' means that the reconstructed tree is the same as the original evolutionary tree. When a reconstructed tree had errors, we checked only how many of the sibling pairs (SPs) were correctly handled.

ISBN: 978-1-61804-350-4

Table 8 Experimental comparisons of the algorithms

Test	CMSM	IPRA	UPGMA	NJ
1	Perfect	Perfect	8 SPs	8 SPs
2	Perfect	Perfect	7 SPs	7 SPs
3	Perfect	Perfect	7 SPs	7 SPs
4	Perfect	Perfect	6 SPs	7 SPs
5	Perfect	Perfect	7 SPs	7 SPs
6	Perfect	Perfect	7 SPs	7 SPs
7	Perfect	Perfect	8 SPs	8 SPs
8	Perfect	Perfect	8 SPs	8 SPs
9	Perfect	Perfect	6 SPs	6 SPs
10	Perfect	Perfect	7 SPs	7 SPs

As an example, Fig. 1 shows the output of the IPRA algorithm in case 4. As a comparison, Fig. 2 shows the output of the UPGMA algorithm in the same case.

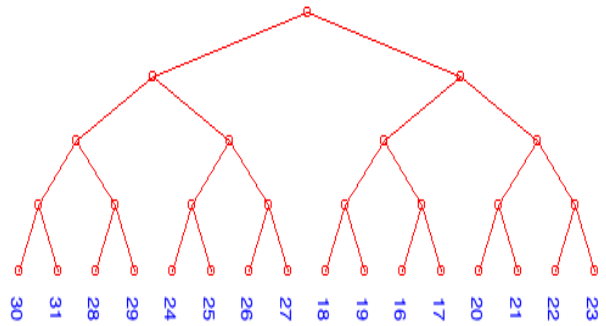

Fig. 1 Sample evolutionary tree reconstructed by the IPRA algorithm

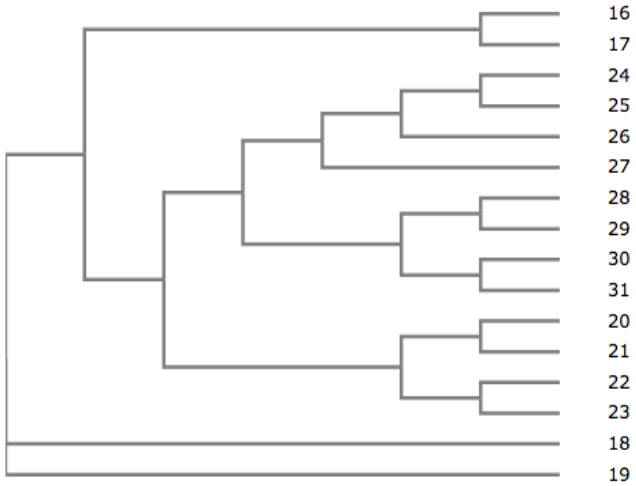

Fig. 2 Sample evolutionary tree reconstructed by the UPGMA algorithm

As can be seen form Figures 1 and 2, the IPRA algorithm has given back the original evolutionary tree. On the other hand, the UPGMA algorithm made a mistake in some of the sibling pairs. In particular, the leaves 26 and 27 and the leaves 18 and 19 are not paired correctly. In addition, there are more mistakes in grouping together cousin leaves. For example, the sibling leaves 16 and 17 are paired correctly, but they are not grouped correctly with their cousin leaves 18 and 19.

V. CONCLUSIONS AND FUTURE WORK

The new incremental phylogenetic tree algorithm has a potential to improve the general of phylogenetic trees and our understanding of evolutionary history, as can be inferred based on molecular biology. Generally all phylogenetic tree algorithms improve with greater data size both in the number of species and in the length of the sequences. In the future, we plan to study additional protein families and their DNA and amino acid sequences. We also plan to develop computer animation software that shows the insertion of new species into the phylogenetic tree.

REFERENCES

[1] D. Baum and S. Smith, *Tree Thinking: An Introduction to Phylogenetic Biology*, Roberts and Company Publishers. 2012.

[2] B. G. Hall, *Phylogenetic Trees Made Easy: A How to Manual*, 4th edition, Sinauer Associates, 2011.

[3] P. Lerney, M. Salemi, and A.-M Vandamme, editors. *The Phylogenetic Handbook: A Practical Approach to Phylogenetic Analysis and Hypothesis Testing*, 2nd edition, Cambridge University Press, 2009.

[4] P. Z. Revesz, *Introduction to Databases: From Biological to Spatio-Temporal*, Springer, New York, 2010.

[5] P. Z. Revesz, "An algorithm for constructing hypothetical evolutionary trees using common mutations similarity matrices," *Proc. 4th ACM International Conference on Bioinformatics and Computational Biology*, ACM Press, Bethesda, MD, USA, September 2013, pp. 731-734.

[6] P. Z. Revesz and C. J.-L. Assi, "Data mining the functional characterizations of proteins to predict their cancer relatedness," *International Journal of Biology and Biomedical Engineering*, 7 (1), 2013, pp. 7-14.

[7] P. Z. Revesz and T. Triplet, "Classification integration and reclassification using constraint databases," *Artificial Intelligence in Medicine,* 49 (2), 2010, pp. 79-91.

[8] P. Z. Revesz and T. Triplet, "Temporal data classification using linear classifiers," *Information Systems*, 36 (1), 2011, pp. 30-41.

[9] N. Saitou and M. Nei, "The neighbor-joining method: A new method for reconstructing phylogenetic trees," *Molecular Biological Evolution*, 4, 1987, pp. 406-425.

[10] M. Shortridge, T. Triplet, P. Z. Revesz, M. Griep, and R. Powers, "Bacterial protein structures reveal phylum dependent divergence," *Computational Biology and Chemistry*, 35 (1), 2011, pp. 24-33.

[11] R. R. Sokal, and C. D. Michener, "A statistical method for evaluating systematic relationships," *University of Kansas Science Bulletin*, 38, 1958, pp. 1409-1438.

[12] T. Triplet, M. Shortridge, M. Griep, J. Stark, R. Powers, and P. Z. Revesz, "PROFESS: A protein function, evolution, structure and sequence database," *Database -- The Journal of Biological Databases and Curation*, 2010, Available: http://database.oxfordjournals.org/content/2010/baq011.full.pdf+html

Peter Z. Revesz holds a Ph.D. degree in Computer Science from Brown University. He was a postdoctoral fellow at the University of Toronto before joining the University of Nebraska-Lincoln, where he is a professor in the Department of Computer Science and Engineering. Dr. Revesz is an expert in databases, data mining, big data analytics and bioinformatics. He is the author of *Introduction to Databases: From Biological to Spatio-Temporal* (Springer, 2010) and *Introduction to Constraint Databases* (Springer, 2002). Dr. Revesz held visiting appointments at the IBM T. J. Watson Research Center, INRIA, the Max Planck Institute for Computer Science, the University of Athens, the

University of Hasselt, the U.S. Air Force Office of Scientific Research and the U.S. Department of State. He is a recipient of an AAAS Science & Technology Policy Fellowship, a J. William Fulbright Scholarship, an Alexander von Humboldt Research Fellowship, a Jefferson Science Fellowship, a National Science Foundation CAREER award, and a "Faculty International Scholar of the Year" award by *Phi Beta Delta*, the Honor Society for International Scholars.

Zhiqiang Li got his B.S. and M.S. degrees from Xidian University, China, in 2009 and 2012, respectively. He is currently a Ph.D. student in the University of Nebraska-Lincoln. His research interests include bioinformatics, computer security, data mining and image processing.

ISBN: 978-1-61804-350-4